

~~Juan Carlos A. Marquez~~
~~Registration Number 34,072~~

PATENT OFFICE

JAPANESE GOVERNMENT

This is to certify that the annexed is a true copy of the following application as filed with this office.

Date of Application : August 11, 2003
Application Number : Patent Application No. 2003-291286
Applicant (s) : Hitachi, Ltd.

Dated this 13th day of February, 2004

Yasuo IMAI
Commissioner,
Patent Office

Certificate No. 2004-3009134

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: 2003年 8月11日

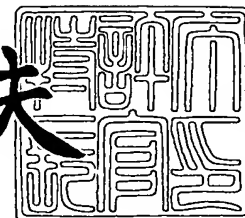
出願番号
Application Number: 特願2003-291286
[ST. 10/C]: [JP2003-291286]

出願人
Applicant(s): 株式会社日立製作所

2004年 2月13日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫



出証番号 出証特2004-3009134

【書類名】 特許願
【整理番号】 GM0306054
【提出日】 平成15年 8月11日
【あて先】 特許庁長官殿
【国際特許分類】 G16F 9/46
【発明者】
 【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地 株式会社日立製作所
 中央研究所内
 【氏名】 増田 峰義
【発明者】
 【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地 株式会社日立製作所
 中央研究所内
 【氏名】 垂井 俊明
【発明者】
 【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地 株式会社日立製作所
 中央研究所内
 【氏名】 樋口 達雄
【特許出願人】
 【識別番号】 000005108
 【氏名又は名称】 株式会社日立製作所
【代理人】
 【識別番号】 100075513
 【弁理士】
 【氏名又は名称】 後藤 政喜
【選任した代理人】
 【識別番号】 100084537
 【弁理士】
 【氏名又は名称】 松田 嘉夫
【選任した代理人】
 【識別番号】 100114236
 【弁理士】
 【氏名又は名称】 藤井 正弘
【手数料の表示】
 【予納台帳番号】 019839
 【納付金額】 21,000円
【提出物件の目録】
 【物件名】 特許請求の範囲 1
 【物件名】 明細書 1
 【物件名】 図面 1
 【物件名】 要約書 1
 【包括委任状番号】 0110326

【書類名】 特許請求の範囲**【請求項 1】**

複数のクライアントと、

前記クライアントからのリクエストを処理する複数のサーバを含み、前記複数のサーバの数を動的に変更するサーバ・クラスタと、によって構成されるクライアント・サーバシステムに用いられる負荷分散方法であって、

前記クライアントは、

前記サーバ・クラスタを構成するサーバの数を検出し、

サーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定し、

前記設定された配分に基づいて前記複数のサーバに対してリクエストを送出することを特徴とする負荷分散方法。

【請求項 2】

前記クライアントは、前記増加したサーバに対して送出されるリクエストの配分を、時間の経過と共に増加するように設定することを特徴とする請求項 1 に記載の負荷分散方法。

【請求項 3】

前記クライアントは、前記サーバ・クラスタのサーバの数の増加が検出されたことを契機として、該増加したサーバに対して送出されるリクエストの配分を、他のサーバに比べて小さく設定することを特徴とする請求項 1 又は 2 に記載の負荷分散方法。

【請求項 4】

前記クライアントは、

前記増加したサーバの性能に関する情報を取得し、

該取得した情報に基づいて、該増加したサーバに対して送出されるリクエストの配分を設定することを特徴とする請求項 1 から 3 のいずれか一つに記載の負荷分散方法。

【請求項 5】

前記クライアントは、

前記増加したサーバの状態に関する情報を取得し、

該取得した情報に基づいて、該増加したサーバに対して送出されるリクエストの配分を設定することを特徴とする請求項 1 から 3 のいずれか一つに記載の負荷分散方法。

【請求項 6】

前記クライアントは、前記サーバの状態に関する情報として、キャッシュヒット率、キャッシュ使用率又はリクエストの待ち数に関する情報の一つ以上を取得することを特徴とする請求項 5 に記載の負荷分散方法。

【請求項 7】

前記クライアント・サーバシステムは、前記サーバの数を管理する管理サーバを備え、

前記クライアントは、前記管理サーバから、前記サーバ・クラスタのサーバの数の増加の通知を受信したことを契機として、該増加したサーバに対して送出されるリクエストの配分を、他のサーバに比べて小さく設定することを特徴とする請求項 1 又は 2 に記載の負荷分散方法。

【請求項 8】

前記クライアント・サーバシステムは、前記サーバの性能に関する情報を取得する管理サーバを備え、

前記クライアントは、

前記管理サーバから、前記増加したサーバの性能に関する情報を取得し、

該取得した情報に基づいて、該増加したサーバに対して送出されるリクエストの配分を設定することを特徴とする請求項 1 又は 2 に記載の負荷分散方法。

【請求項 9】

前記クライアントは、前記サーバとの間の通信接続数を設定することによって、前記増加したサーバに対して送出されるリクエストの配分を設定することを特徴とする請求項 1

から 8 のいずれか一つに記載の負荷分散方法。

【請求項 1 0】

前記クライアントは、前記サーバに送出されるリクエストの各サーバに対する割当を変更することによって、前記各サーバに対して送出されるリクエストの配分を設定することを特徴とする請求項 1 から 8 のいずれか一つに記載の負荷分散方法。

【請求項 1 1】

前記クライアント・サーバシステムは、前記サーバに接続されるストレージ装置を備え、
前記サーバは、前記ストレージ装置に記憶されるファイルの格納場所を示すディレクトリ情報を保持し、

前記クライアントは、前記サーバに送出されるリクエストの各サーバに対する割当として、各サーバへの前記ディレクトリ情報を格納する割当を変更することによって、前記各サーバに対して送出されるリクエストの配分を設定することを特徴とする請求項 1 0 に記載の負荷分散方法。

【請求項 1 2】

複数のクライアントと、
前記クライアントからのリクエストを処理する複数のサーバを含み、前記複数のサーバの数を動的に変更するサーバ・クラスタと、によって構成されるクライアント・サーバシステムであって、
前記クライアントは、
前記各サーバに対して送出されるリクエストの配分を設定する負荷設定部と、
前記サーバ・クラスタを構成するサーバの数を検出する台数検出部と、
前記負荷設定部によって設定された配分に基づいて、前記複数のサーバに対してリクエストを送出する負荷分散部と、を備え、

前記負荷設定部は、前記台数検出部によってサーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定することを特徴とするクライアント・サーバシステム。

【請求項 1 3】

前記クライアントは、前記サーバに送出されるリクエストの各サーバに対する割当てを保持する割当保持部を備え、

前記負荷分散部は、前記リクエストの前記サーバに対する割当を変更することによって、前記各サーバに対して送出するリクエストの配分を設定することを特徴とする請求項 1 2 に記載のクライアント・サーバシステム。

【請求項 1 4】

前記クライアント・サーバシステムは、前記サーバに接続されるストレージ装置を備え、
前記サーバは、前記ストレージ装置に記憶されるファイルの格納場所を示すディレクトリ情報を保持するディレクトリ情報保持部を備え、

前記クライアントは、前記サーバに送出されるリクエストの各サーバに対する割当として、前記ディレクトリ情報を格納しているサーバの割当を保持する割当管理部を備え、

前記負荷分散部は、前記ディレクトリ情報を格納しているサーバの割当を変更することによって、前記各サーバに対して送出するリクエストの配分を設定することを特徴とする請求項 1 3 に記載のクライアント・サーバシステム。

【書類名】 明細書

【発明の名称】 負荷分散方法及びクライアント・サーバシステム

【技術分野】

【0001】

本発明は、インターネット上での電子商取引等のサービス構築に用いられる、サービス利用者からのリクエストを処理するサーバ・クラスタシステム等を用いたクライアント・サーバシステムにおいて、サービス需要の増減に応じてサーバ・クラスタシステムを構成するサーバの台数を変更するクラスタ再構成技術に適合した、クライアントとサーバ・クラスタシステム間の負荷分散方法及び負荷分散方法を実装したクライアント・サーバシステムに関する。

【背景技術】

【0002】

複数のサーバをネットワークで結合し、見かけ上単一のサーバを構成するサーバ・クラスタシステムが、インターネット上で電子商取引等の各種サービスを提供する計算機システムに用いられている。計算機システムとしてサーバ・クラスタシステムを用いる場合、クライアントとサーバ・クラスタシステムとの間で負荷分散を行うことが一般的である。具体的には、クライアントからのサービス処理要求（リクエスト）を、サーバ・クラスタシステムを構成する複数のサーバの各々の処理能力に応じて振り分けるよう、リクエストの配分を決定する。

【0003】

この負荷分散に用いるアルゴリズム、すなわち、どのサーバにどれだけのリクエストを送出するかを決定する手順は、システムの性能を左右する。負荷分散アルゴリズムが適切でない場合、クライアントからのリクエストが均等にサーバに振り分けられず、サーバ間で負荷の不均衡が生じる。負荷の重いサーバでは、負荷が軽い他のサーバと比較すると、リクエストに対する処理時間が大幅に大きくなり、リクエスト処理が遅れてしまう。例えば、インターネット上でのサービスの場合、リクエスト処理の遅れは、サービス利用者への応答の遅れとして顕在化する。

【0004】

この、システムにとって重要な要素である負荷分散アルゴリズムの代表的なものにラウンドロビン方式が知られている。ラウンドロビン方式とは、同時に入力する複数の要求信号の内の一つを予め設定された優先順位に従って、かつ均等に選択されるように、優先順位の並び替えを行う方式である。このラウンドロビン方式を利用すると、負荷分散先のサーバに、リクエストを順番に送出手続きができる（例えば、特許文献1参照。）。

【0005】

また、ラウンドロビン方式を応用した方式に、重み付きラウンドロビン方式がある。重み付きラウンドロビン方式とは、ラウンドロビン方式の優先順位を、所定の重みに従って並び替えを行う方式である。この重み付けラウンドロビン方式を利用すると、サーバのプロセッサの動作周波数やメモリ量などを基に、サーバの性能値を“重み”として算出しておき、サーバに、重みに応じた数のリクエストを振り分けることができる（例えば、特許文献2参照。）。

【0006】

また、近年、クラスタ再構成技術と呼ばれる、サーバ・クラスタシステムを動的に再構成する技術が提案されている。このクラスタ再構成技術は、クライアントからのサービス要求を、負荷分散装置によってサーバ・クラスタで稼働中のいずれかのサーバに振り分け、アクセスの負荷に応じてサーバ・クラスタの構成（台数）を変更する技術である。このクラスタ再構成技術は、インターネット上でのサービスを提供するシステムのように、サービス利用者数の変動が大きいシステムでは特に有効である（例えば、特許文献3参照。）。

【0007】

また、NAS（Network Attached Storage）にサーバ・クラスタシステムを適用し、容

量拡張、減縮が容易に行えるよう、ファイルの識別子にハッシュ関数を適応し、ファイルと該ファイルの格納先とを組み合わせたテーブルを管理するシステムも知られている（例えば、非特許文献1参照。）。

【特許文献1】特開2000-231496号公報

【特許文献2】特開2001-217836号公報

【特許文献3】特開2002-163241号公報

【非特許文献1】川本 真一，他5名，“ファイル自律配置方式を備えた仮想一元化NASシステムX-NASの実現と評価”，[online]，[平成15年7月17日検索]，インターネット<URL：<http://www.ieice.org/iss/de/DEWS/proc/2003/papers/4-B/4-B-01.pdf>>

【発明の開示】

【発明が解決しようとする課題】

【0008】

前述したクラスタ再構成技術を、不適切な負荷分散アルゴリズムと併用した場合、サーバ・クラスタシステムに新たにサーバを追加した際に、サーバ・クラスタシステム内を構成するサーバ間で負荷の不均衡が発生することがある。このサーバ間での負荷の不均衡は、クラスタ再構成技術を用いてサーバ・クラスタシステムに新たに追加したサーバと、クラスタ内で既に動作していた既存のサーバとの間で、リクエスト処理能力に差があることが原因である。

【0009】

このような、サーバ間でリクエスト処理能力に差が起こる原因の一つに“キャッシュ”がある。

【0010】

例として、複数のキャッシュ・サーバで構成されるサーバ・クラスタシステムについて説明する。キャッシュ・サーバは、Webサーバの前段（Webサーバとサービス利用者との間）に配置され、Webサーバの代わりに、サービス利用者からのWebコンテンツ配信要求に対するWebコンテンツを配信する。要求されたWebコンテンツがキャッシュ・サーバ内にキャッシングされている場合、キャッシングされているコンテンツをWebサーバから取得することなく、サービス利用者に即時に配信できるため、サービス利用者への応答時間は短い。一方、Webコンテンツがキャッシュ・サーバ内にキャッシングされていない場合、Webコンテンツを、一旦Webサーバから取得した後、サービス利用者に配信を行う必要があるため、サービス利用者への応答時間が長くなる。つまり、キャッシュ・サーバ内にWebコンテンツがキャッシュされているか否かで、サービス利用者への応答時間に大きな差ができる。

【0011】

このような特性を持つキャッシュ・サーバから構成されるサーバ・クラスタシステムに、新たにキャッシュ・サーバを一台追加した場合、キャッシュ・サーバを追加した直後は、追加されたキャッシュ・サーバ内にはWebコンテンツは全くキャッシングされていない。そのため、追加直後のキャッシュ・サーバは、利用者からの要求を受けるたびにWebサーバからコンテンツを取得する必要があるため、サービス利用者への応答時間が長くなる。一方、これまでサーバ・クラスタシステム内で稼動していた既存のキャッシュ・サーバは、多くのWebコンテンツをキャッシングしているので、新たに追加されたキャッシュ・サーバと比較すると、サービス利用者への応答時間が短い。

【0012】

このように、サービス利用者からの要求を処理する能力に差がある既存キャッシュ・サーバと追加キャッシュ・サーバに、ラウンドロビン方式で同数のリクエストを送出した場合、追加されたキャッシュ・サーバでは、リクエストの処理が間に合わず、サーバ内で未処理リクエストの長い待ち行列が発生する。そのため、待ち行列の理論に従い、サービス利用者への応答時間は指数関数的に長くなる。そのため、応答時間の大幅な遅れが発生し、サービス利用者に多大な不利益を与える、ひいてはサービス提供者の信頼を損なう。

【0013】

本発明は、このような問題に鑑みてなされたものであり、サーバ・クラスタシステムに新たにサーバが追加された際、追加されたサーバにおいて、リクエスト処理時間が長大になることを回避する分散制御方法を提供する。

【課題を解決するための手段】**【0014】**

本発明は、複数のクライアントと、前記複数のクライアントからのリクエストを処理する複数のサーバを含み、前記複数のサーバの数を動的に変更するサーバ・クラスタと、によって構成されるクライアント・サーバシステムに用いられる負荷分散方法であって、前記クライアントは、前記サーバ・クラスタを構成するサーバの数を検出し、サーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定し、前記設定された配分に基づいて前記複数のサーバに対してリクエストを送出することを特徴とする。

【0015】

本発明では、サーバ・クラスタシステムにサーバを追加した直後は特別な負荷分散を行う。すなわち、既存のサーバと比較して性能（処理能力）が劣る追加サーバに対するリクエスト送出量を、既存サーバに比べて抑えることで、追加サーバにおいてリクエスト処理時間が長大になることを回避する。

【0016】

本発明のクライアント・サーバシステムは、クライアントからのリクエストを処理するための複数のサーバから構成されるサーバ・クラスタシステムとクライアントとからなり、サーバ・クラスタシステムを構成するサーバの台数は、リクエスト数の変動に応じて増減する。クライアントには、サーバ・クラスタシステムにリクエストを振り分ける負荷分散機能と、リクエストの振り分け方を制御する負荷制御プログラムとを備える。また、前記負荷制御プログラムはサーバ・クラスタシステムのサーバ台数の変更を検出する機能を持つ。

【0017】

本発明の特徴は、負荷制御プログラムが追加サーバへのリクエスト送出量を調整し、既存サーバへのリクエスト送出量に比べて、追加直後の追加サーバへのリクエスト送出量を少なくし、時間経過と共に追加サーバへのリクエスト送出量を段階的に増加させることである。

【0018】

負荷制御プログラムは、サーバ・クラスタシステムを構成するサーバ台数の変更を検知する機能を持つ。例えば、サーバ・クラスタシステムのサーバ台数を管理する管理サーバを備え、管理サーバが、サーバ台数の変更があった場合に、その変更を負荷制御プログラムに通知する。負荷制御プログラムは、サーバ台数変更の通知の検出を契機として、追加サーバへのリクエストの送出を開始する。

【0019】

負荷制御プログラムは、追加サーバへのリクエスト送出量を段階的に増加させる。この増分の計算方法には二つの方式がある。一つは、追加サーバから性能情報を取得し増分計算に利用する方式、もう一つは、サーバから性能情報を取得しない方式である。

【0020】

サーバ・クラスタシステムからの性能情報を利用する場合、負荷制御プログラムは、追加サーバから、例えばキャッシュヒット率やリクエストの待ち行列長といった性能情報を取得する機能と、その性能情報を基にリクエスト送出量の増分を計算する機能を持ち、計算結果に従って、追加サーバへのリクエスト送出量を増加させる。

【0021】

追加サーバからの性能情報を利用しない場合、負荷制御プログラムは、予め定められた規則（例えば、サーバが追加されてから所定の時間内は、リクエストの送出量を10秒間毎に10%増加させる）に従い追加サーバへのリクエスト送出量を増加させる。

【発明の効果】**【0022】**

本発明によると、複数のクライアントと、前記複数のクライアントからのリクエストを処理する複数のサーバを含み、前記複数のサーバの数を動的に変更するサーバ・クラスタと、によって構成されるクライアント・サーバシステムに用いられる負荷分散方法であって、前記クライアントは、前記サーバ・クラスタを構成するサーバの数を検出し、サーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定し、前記設定された配分に基づいて前記複数のサーバに対してリクエストを送出することを特徴とするので、サーバ・クラスタシステムに新たにサーバが追加された際に、追加サーバでのリクエスト処理時間が長大となることを回避できる。

【発明を実施するための最良の形態】**【0023】**

本発明の実施の形態について、図面を参照して説明する。

【0024】

図1は、本発明の第1の実施の形態のクライアント・サーバシステムの概要を表すブロック図である。

【0025】

本実施の形態のクライアント・サーバシステムは、複数のクライアント100がサーバ間ネットワーク700によってサーバ・クラスタシステム1100と接続されている。

【0026】

クライアント100は、サーバ・クラスタシステム1100にサービスを要求し、サーバ・クラスタシステム1100から要求に対する結果を受け取る。このクライアント100では、負荷分散機能300及び負荷制御プログラム400が動作している。なお、クライアント100からサーバ・クラスタシステム1100に送信するサービスの処理の要求を「リクエストの送出」と呼ぶ。

【0027】

負荷制御プログラム400は、サーバ・クラスタシステム1100を構成する各サーバに対して、各々どれだけの配分でリクエストを送出するのかをクライアント100が判断するためのデータである、負荷重み表405を作成する。また、負荷分散機能300は、負荷制御プログラム400が作成した負荷重み表405に従って、各々のサーバとクライアントプログラム200との間の通信量を制御する。すなわち、サーバ・クラスタシステム1100を構成する各サーバに対するリクエストの送出を振り分け、各サーバにリクエストを送出する。

【0028】

このように、負荷制御プログラム400が各サーバに対するリクエストの配分を決定することによって負荷設定部が構成される。

【0029】

サーバ・クラスタシステム1100は、複数のサーバがネットワークによって疎結合されており、クライアント100からは単一のサーバとして見えるシステムである。

【0030】

このサーバ・クラスタシステム1100には、クラスタ再構成技術が適用され、サービスを停止させることなく、システムを構成するサーバの台数を変更することができる。このクラスタ再構成技術は、サーバ・クラスタシステム1100に新たなサーバの追加、及びサーバ・クラスタシステム1100で動作しているサーバの削減ができる。なお、サーバ・クラスタシステム1100に追加された新たなサーバを「追加サーバ900」、既にサーバ・クラスタシステム内で動作しているサーバを「既存サーバ800」と呼ぶ。

【0031】

また、クライアント100及びサーバ・クラスタシステム1100には、サーバ・クラスタシステム1100のサーバやサービスの構成の状態を取得し管理する、管理サーバ6

00が接続されている。

【0032】

サーバ・クラスタシステム1100は、管理サーバ600とエージェント・プログラム1000とによってクラスタ再構成技術が実装されている。具体的には、エージェント・プログラム1000は、サーバ・クラスタシステム1100を構成する全てのサーバに備えられ動作している。管理サーバ600とエージェント・プログラム1000とは相互に通信しながらサーバ・クラスタシステム1100の構成を管理する。エージェント・プログラム1000は、サーバ・クラスタシステム1100を構成する各サーバの負荷を計測し、定期的に管理サーバ600に報告する。管理サーバ600は、エージェント・プログラム1000より報告を受け収集した負荷情報を分析し、サーバ・クラスタシステム1100へのサーバの追加、又はサーバ・クラスタシステム1100からのサーバ削減を決定する。

【0033】

なお、管理サーバ600を設けずに、管理サーバ600の機能を、サーバ・クラスタシステム1100に組み込んだり、クライアント100に組み込んだりもよい。

【0034】

本発明の第1の実施の形態では、サーバ・クラスタシステム1100に新規に追加された追加サーバ900へのリクエスト送出量を、追加した当初（直後）は既存サーバ800に比べて低く抑え、その後は段階的に増やしていくように制御する。追加サーバ900がサーバ・クラスタシステム1100に追加された直後は、追加サーバ900に備えられているキャッシュメモリ等に十分なキャッシュが蓄えられておらず、追加サーバ900の処理能力は既存サーバ800と比較すると低いので、既存サーバ800と同等のリクエストが送出されると、未処理リクエストの長い待ち行列が発生してしまう可能性がある。そのため、追加サーバ900がサーバ・クラスタシステム1100に追加された直後は、クライアント100からのリクエスト送出量を低く抑えるように制御する。その後、所定の時間が経過すればキャッシュメモリ等に十分なキャッシュが蓄えられ、既存サーバ800と同様の処理を行うことができるようになる。

【0035】

この制御は、負荷制御関数411を適切に定義することで実現できる。すなわち、追加された直後のサーバの重みを、初めは小さな値に、時間経過と共に段階的に大きな値になるように算出する負荷制御関数411を定義する。

【0036】

本発明のクライアント・サーバシステムの実施形態の一例として、3層Webシステムが挙げられる。3層Webシステムは、Webサーバ、アプリケーション（AP）サーバ、データベース（DB）サーバが、それぞれサーバ・クラスタシステムを構成し、かつ、それらのサーバ・クラスタシステムがWebサーバ、APサーバ、DBサーバの順に層構造をなすシステムである。3層Webシステムでは、サービス利用者からのリクエストは、階層構造に従って、Webサーバ→APサーバ→DBサーバの順に流れ作業で処理される。この3層Webシステムでは、サービス利用者が使用する計算機とWebサーバの間、例えば、WebサーバとAPサーバの間、APサーバとDBサーバとの間がクライアントとサーバの関係となる。具体的には、WebサーバとAPサーバとの間では、WebサーバがクライアントでありAPサーバがサーバである。また、APサーバとDBサーバとの間では、APサーバがクライアントであり、DBサーバがサーバである。

【0037】

また、クライアント100がWebサーバであり、Webサーバ・プログラムとしてApacheを用い、サーバ・クラスタシステム1100がAPサーバであり、APサーバ・プログラムとしてTomcatを用いた場合、負荷分散機能300は、Apacheに組み込まれるTomcatの負荷分散モジュールとなる。

【0038】

次に、負荷制御プログラム400の詳細について説明する。

【0039】

図2は、負荷制御プログラム400構造及び処理の流れを示すブロック図である。

【0040】

負荷制御プログラム400は、サーバ性能表403と負荷重み表405の二つのデータをテーブルとして持つ。また、サーバ性能表403を作成するための機能である台数検出機能401及び性能検出機能402を持つ。更に、サーバ性能表403から負荷重み表405を作成する機能である負荷重み計算機能404を持つ。

【0041】

403は、サーバ性能表の一例を示している。サーバ性能表403には、サーバ・クラスタシステム1100を構成するサーバ毎にエントリが作成される。各エントリには、サーバ性能表403内でのエントリ管理番号としてサーバ毎に付されたサーバ番号（サーバ#）、サーバにアクセスを行うためのURL（Uniform Resource Locator）情報（URL）、サーバの性能を示す複数のパラメータ（P1～PN）、サーバが新たに追加された時刻（t0）が記録されている。

【0042】

サーバの性能を示すパラメータの例としては、サーバのプロセッサの個数、プロセッサの動作周波数、主記憶搭載量等のシステム稼動中には変更が行われない静的パラメータ、及び、サーバ内でのキャッシュヒット率、サーバ内でのリクエスト待ち行列長等のシステム稼動中に処理の負荷状況に応じて変更される動的パラメータの2種類が記録される。

【0043】

405は、負荷重み表の一例を示している。負荷重み表405は、サーバ性能表403と同じく、サーバ・クラスタシステム1100を構成するサーバ毎にエントリが作成され記録されている。各エントリには、負荷重み表405内でのエントリ管理番号としてサーバ毎に付されたサーバ番号（サーバ#）、サーバにアクセスを行うためのURL、サーバ性能の評価値である重みが記録されている。

【0044】

台数検出機能401及び性能検出機能402の役割は、サーバ性能表403のエントリを作成し記録することである。台数検出機能401は、サーバ性能表403のエントリを新規に作成し、サーバの静的パラメータを記録する。性能検出機能402はサーバの動的パラメータを取得し記録する。

【0045】

負荷重み計算機能404は、所定のタイミングに動作し、サーバ性能表403を基に負荷重み表405の内容を更新する。具体的には、負荷重み計算機能404は、一定時間間隔で定期的に動作し、サーバ性能表403から各サーバの性能パラメータを読み込んで、読み込んだ性能パラメータを負荷制御関数411に入力する（メッセージ408）。負荷制御関数411は読み込んだ各サーバの性能パラメータから各サーバの重みを計算し、計算した重みを負荷重み表405に記録する（メッセージ409）。

【0046】

なお、この負荷制御関数411は自由に設定し記述することが可能であり、サーバ性能の動的パラメータを入力値とした複雑な多次元関数として定義することもできる。例えば、キャッシュのヒット率が低い場合はリクエスト送出量を低くする関数、リクエストの待ち行列長が高い場合にはリクエストの送出量を低くする関数等を負荷制御関数411として設定する。また、複数の負荷制御関数411を予め設定しておき、条件によって複数の負荷制御関数411を切り替えて使い分けることもできる。

【0047】

この負荷制御関数411の例を図3及び図4に示す。図3の例では、負荷制御関数411を、サーバが新たに追加されてからの時間（サーバ性能表403のパラメータ“t0”）を入力値として関数を定義している。この負荷制御関数には、サーバが追加されてから所定の時間が経過するまではリクエスト送出量を時間に比例して増加させ、所定の時間が経過した後は入力値（経過時間）に関係なくリクエスト送出量を一定とする関数が定義さ

れている。

【0048】

図4の例では、負荷制御関数411を、待ち行列長を入力値として関数を定義している。図4によると、待ち行列長とリクエスト送出量とが反比例の関係となるような関数が定義されている。すなわち、待ち行列長が長い間はリクエスト送出量を減少させ、待ち行列長が短くなるとリクエスト送出量を増加させる。

【0049】

図5は、負荷制御プログラム400の台数検出機能401の処理を示すフローチャートである。なお、図5では、サーバ・クラスタシステム1100に新たにサーバ（追加サーバ900）が追加された場合を例として示している。

【0050】

まず、台数検出機能401は通常は待機状態にある。ここで、管理サーバ600からサーバ・クラスタシステム1100の構成（サーバの数等）が変更されたことの通知を受信すると（処理1201）、該通知から追加サーバ900に関する情報を取得する（処理1202）。

【0051】

次に、管理サーバ600から受信した追加サーバ900に関する情報から、追加サーバ900のサーバ名を取得する（処理1203）。次に、取得したサーバ名を用いて、サーバ性能表403内に該追加サーバ900のエントリが存在するかを判定する（処理1204）。エントリが無いと判定した場合は、追加サーバ900に対する一意なサーバ番号を新規に生成する。そして、生成したサーバ番号のエントリをサーバ性能表403に新たに作成し（処理1205）、処理1206に進む。既にエントリがあると判定した場合は、エントリの新規作成は行わず処理1206に進む。

【0052】

処理1206では、受信した追加サーバ900に関する情報から、追加サーバ900のURL、追加サーバ900の静的なパラメータを抽出して取得し、サーバ性能表403の該当するサーバ番号のエントリに書き込んで記録する。その後、再び待機状態に戻り、構成が変更されたことの通知を待機する。

【0053】

このように、台数検出機能401が追加サーバの数に変更されたことを検出することによって台数検出部が構成される。

【0054】

図6は、負荷制御プログラム400の、性能検出機能402の処理を示すフローチャートである。

【0055】

性能検出機能402は所定のタイミングに動作し、サーバ性能表403の各エントリの動的パラメータの値を更新する。まず、サーバ性能表403内のエントリを調査し、現在サーバ性能表403に記録されているサーバ名を取得する（処理1301）。次いで、サーバ性能表403に記録されている各サーバと通信し、動的パラメータに関する性能情報を取得する（処理1302）。次に、取得した性能情報をサーバ性能表403に記録する（処理1303）。そして、この処理1301から処理1303の処置をエントリの数だけ繰り返す。

【0056】

このように、性能検出機能401が、各サーバの状態である動的パラメータを取得することによって状態取得部が構成される。

【0057】

図7は、負荷重み計算機能404の処理を示すフローチャートである。

【0058】

負荷重み計算機能404は、所定のタイミング（例えば、タイマのカウント等によって定期的）にサーバ性能表403を参照し、サーバ性能表403から所定のエントリのサー

バの性能パラメータを取得する(処理1401)。次に、取得した性能パラメータを入力値として負荷制御関数411に入力し、該エントリのサーバの重みを計算する(処理1402)。次に、計算した重みを、負荷重み表405の当該サーバのエントリに書き込み記録する(処理1403)。次に、サーバ番号を加算することで次のエントリを設定する(処理1404)。このようにすることで、サーバ性能表403のサーバ番号の順に、全てのエントリに対する重みが算出される。

【0059】

なお、この図7の負荷重み計算機能404の処理の実行タイミングは、定期的に行う以外に、台数検出機能401又は性能検出機能402がサーバ性能表403のエントリを更新したことを契機に行うことができる。例えば、台数検出機能401が、サーバ性能表403のエントリを更新した後、その旨を負荷重み計算機能404に通知する。負荷重み計算機能404は、この通知を契機として図7のフローチャートに従って重みを算出し、負荷重み表405を更新する。性能検出機能402がサーバ性能表403を更新した場合も、負荷重み計算機能404に対して同様の通知を行うことで重みを算出し、負荷重み表405を更新する。

【0060】

次に、負荷分散機能300によって、クライアントプログラム200とサーバ・クラスタシステム1100との間で行われる負荷分散の処理について説明する。

【0061】

まず、コネクション・プールについて説明する。

【0062】

コネクション・プールとは、計算機間通信の高速化技術である。一般に、計算機間で通信を行う場合、「コネクション」と呼ばれる通信路を確立する必要がある。なお、確立されたコネクションは通信が終了すると破棄される。このコネクションを確立する処理には時間を要するため、通信の度にコネクションを確立する処理をしていては通信効率が低下する。コネクション・プールは、一度確立したコネクションを、使用後(通信終了後)にも廃棄せずにコネクションの状態を記憶(プール)する。プールされたコネクションは、再度、同一経路での通信が行われる際に再利用することで、再度コネクションを確立する処理を省略することができ、通信効率を向上することができる。

【0063】

図8は、三台のサーバ(800a、800b、800c)で構成されたサーバ・クラスタシステム1100にクライアントプログラム200がリクエストを送出する場合のコネクション・プールの実装例を示す。

【0064】

コネクション配分機能301は、各々のサーバにコネクション・プール304を作成し、その管理を行う。

【0065】

図8は、クライアント100からサーバ800aへは、五つのコネクション・プール304aのうち三つのコネクション(図8中の網掛け部分)を使用し、クライアント100からサーバ800bへは五つのコネクション・プール304bのうち一つのコネクションを使用し、クライアント100からサーバ800cへは五つのコネクション・プール304cのうち二つのコネクションを使用している状況を表している。

【0066】

なお、図8では全てのサーバに対して同じ数のコネクション・プールを設けているが、プール可能なコネクションの数はサーバ毎に異なっても構わない。

【0067】

コネクション配分機能301は、コネクション管理表302に、プールしているコネクションの数と現在使用中のコネクションの数とを、コネクション・プール304毎に記録する(メッセージ303)。

【0068】

クライアントプログラム 200 は、サーバ・クラスタシステム 1100 にリクエストを送出する際、通信に必要なコネクションをコネクション・プールから取得して使用するために、まずコネクション配分機能 301 に対して、コネクションの割当てを要求する（メッセージ 201）。要求を受けたコネクション配分機能 301 は、まずコネクション管理表 302 を参照し（メッセージ 303）、どのコネクション・プール 304 からコネクションを取得するかを決定する（図 8 では、コネクション・プール 304 b から取得）。次に、取得したコネクションをクライアントプログラム 200 に割当て、その旨をクライアントプログラム 200 に送信する（メッセージ 202）。クライアントプログラム 200 は、割当てられたコネクションを使用してサーバ 800 b と通信を行う。クライアントプログラム 200 は、通信が終了すると、コネクションの使用が終了した旨のメッセージ 203 をコネクション配分機能 301 に送信して、コネクションをコネクション配分機能 301 に返却する。コネクション配分機能 301 は、このメッセージを受け取って、使用中のコネクションを未使用のコネクションに変更して、コネクション管理表 302 を更新する。

【0069】

図 9 は、負荷分散機能 300 に前述したコネクション・プールを用いた場合の、負荷分散機能のデータ構造、及びデータの処理を示すブロック図である。

【0070】

負荷分散機能 300 は、クライアントプログラム 200 とサーバ・クラスタシステム 1100 との間の負荷分散を行う。負荷分散機能 300 は、コネクション配分機能 301 とコネクション管理表 302 を保持している。

【0071】

クライアントプログラム 200 は、サーバ・クラスタシステム 1100 にリクエストを送出する際に、通信に必要なコネクションの割当てを要求する（メッセージ 201）。要求を受けたコネクション配分機能 301 は、コネクション管理表 302 を参照して（メッセージ 303）、どのコネクション・プールからコネクションを取得するかを決定する。

【0072】

コネクション管理表 302 は、図 9 に示すように、サーバ番号（サーバ#）、サーバにアクセスするための URL、プール可能なコネクション数の最大の値（最大コネクション）、現在使用されているコネクションの数（使用中コネクション）を一覧として保持している。

【0073】

コネクション配分機能 301 は、このコネクション管理表 302 を参照して、どのサーバにいくつのコネクションを割当てるかを決定し、結果をコネクション管理表 302 に記録して反映する。クライアントプログラム 200 は、割当てられたコネクションを使用してサーバにリクエストを送出する。

【0074】

図 10 は、負荷分散機能 300 の処理を示すフローチャートである。

【0075】

負荷分散機能 300 は、クライアントプログラム 200 からコネクション割当ての要求を受信すると（処理 1501）、コネクション管理表 302 を参照し、現在のコネクション割当て状況に関する情報を取得する（処理 1502）。次に、負荷重み表 405 を参照し、各サーバの重みを取得する（処理 1503）。なお、処理 1502 と処理 1503 の順序は逆でもよい。

【0076】

次に、コネクション配分機能 301 が、取得した現在のコネクション割当て状況とサーバ毎の重みとを基に、重み付きラウンドロビン等の負荷分散アルゴリズムに従って、どのサーバにいくつのコネクションを割当てるかを選定する（処理 1504）。次に、選定したコネクションを、クライアントプログラム 200 に通知し、各サーバに対するコネクションを割当て（処理 1505）。

【0077】

このように、コネクション・プールを使用することで、サーバ・クラスタシステム 1 1 0 0 を構成する各サーバの負荷に応じて、各サーバのコネクションの割当てを決定する。各サーバに割当てたコネクションの数に従って、リクエスト送出手の割合が決まる。

【0078】

以上説明したように、本発明の第 1 の実施の形態のクライアント・サーバシステムでは、クライアント 1 0 0 がリクエストを送出するサーバ・クラスタシステム 1 1 0 0 の構成が変更され、新たに追加サーバ 9 0 0 が追加されたときは、該追加サーバ 9 0 0 に対するリクエストの送出量を、既存サーバ 8 0 0 と比較して少なくする。このようにすることで、追加サーバ 9 0 0 の、未処理リクエストの長い待ち行列の発生を抑えることができ、サーバ・クラスタシステム 1 1 0 0 全体としての処理の効率を高めることができる。

【0079】

なお、第 1 の実施の形態では、性能検出機能 4 0 2 が取得する動的パラメータの例として、キャッシュヒット率、待ち行列長を挙げたが、その他に、メモリ使用量、スワップ回数等のメモリに関連する情報、CPU 使用率等の CPU に関連する情報、物理ディスクへの入出力量等の入出力に関連する情報、ネットワーク通信量等のネットワークに関連する情報等を動的パラメータに使用して、リクエスト送出量を制御してもよい。

【0080】

また、負荷制御関数 4 1 1 への入力値として、動的パラメータの値をそのまま入力したが、値の変化量を負荷制御関数 4 1 1 に入力してもよい。例えば、取得した待ち行列長の値が 1 0 から 3 0 に変化した場合、入力値を 3 0 とするのではなく、変化量の 2 0 を入力値とする。変化量を入力することで、待ち行列長が短期間に急激に上昇した場合に、重みを小さくする制御が可能となる。なお、変化量を算出するためには、以前に取得した値を、負荷重み表 4 0 5 又は別の表に記録しておく必要がある。

【0081】

また、性能検出機能 4 0 2 は、サーバ・クラスタシステム 1 1 0 0 に属するサーバから直接性能情報 5 0 0 を取得したが、その他の取得方法として、管理サーバ 6 0 0 がサーバ・クラスタシステム 1 1 0 0 に属するサーバから性能情報 5 0 0 を取得し、その後、性能検出機能 4 0 2 へ性能情報 5 0 0 をまとめて送信してもよい。

【0082】

また、負荷分散機能 3 0 0 及び負荷制御プログラム 4 0 0 は、各クライアント 1 0 0 内に実装されているが、負荷分散装置を別に設け、負荷分散装置によって、複数のクライアント 1 0 0 からのリクエストを集約し、サーバ・クラスタシステム 1 1 0 0 の各サーバにリクエストを振り分けるように実装してもよい。

【0083】

次に、本発明の第 2 の実施の形態について説明する。

【0084】

第 2 の実施の形態は、本発明をストレージ・システム（ストレージ装置）に適用した場合の実施形態であり、クライアント、ディレクトリサーバ・クラスタシステム及びストレージによって、サーバ・クライアントシステムが構成されている。

【0085】

図 1 1 は、本発明の第 2 の実施の形態のストレージ・システムの構成を示したブロック図である。

【0086】

本実施の形態のストレージ・システムは、ディレクトリサーバ・クラスタ 2 6 0 0 が、クライアント 2 0 0 0 にファイルサービスを提供する。

【0087】

クライアント 2 0 0 0 は、ディレクトリサーバ・クラスタ 2 6 0 0 に対してファイルサービスを要求する。ディレクトリサーバ・クラスタ 2 6 0 0 は、複数のディレクトリサーバ（既存ディレクトリサーバ 2 1 0 0 及び追加ディレクトリサーバ 2 2 0 0）がネットワ

ークによって疎結合されており、クライアント2000からは単一のディレクトリサーバとして見えるクラスタシステムである。

【0088】

ファイルの実体（データ）はストレージ2300に格納されている。ディレクトリサーバ・クラスタ2600とストレージ2300とは、SAN（Storage Area Network）2500によって接続されている。ディレクトリサーバ・クラスタ2600とSAN2500とはネットワーク2501を介して接続され、ストレージ2300とSAN2500とはネットワーク2502を介して接続されている。また、クライアント2000とディレクトリサーバ・クラスタ2600とは、サーバ間ネットワーク2400によって接続されている。

【0089】

一般的なファイルシステムは、ファイルの格納場所を指示する「ディレクトリ情報」と、ディレクトリ情報によって指示される「ファイルの実体」とからなり、ディレクトリ情報とファイルの実体（データ）は同一の記憶装置に格納される。

【0090】

本実施の形態のストレージ・システムでは、ディレクトリ情報はディレクトリサーバ・クラスタ2600に格納され、ファイルの実体はストレージ2300に格納される。そのため、ディレクトリ情報には、ファイルが格納されているストレージを示す情報と、該ストレージ内での格納位置を示す情報が記録される。

【0091】

これら複数のディレクトリサーバ（既存ディレクトリサーバ2100及び追加ディレクトリサーバ2200）はクラスタ構成をとっており、ディレクトリ情報は複数のディレクトリサーバ2200に分散して格納される。なお、あるファイルのディレクトリ情報を格納しているディレクトリサーバ2100を、そのファイルの「担当ディレクトリサーバ」と呼ぶ。あるファイルとそのファイルの担当ディレクトリサーバとの対応は、クライアント2000が保持するファイル割当て管理表2001に記録されている。なお、全てのクライアント2000は、同一内容のファイル管理表2001を持っている。

【0092】

このように、クライアント2000が、あるファイルとそのファイルの担当ディレクトリサーバとの対応を割当て管理表2001が保持することによって、割当て保持部が構成される。

【0093】

次に、クライアント2000が出すファイル取得要求を、このストレージ・システムが処理する手順について説明する。

【0094】

クライアント2000は、ストレージ・システムのディレクトリサーバ・クラスタ2600にファイル取得要求（メッセージ2401）を送信する。クライアント2000がファイルを取得するためには、まず、取得を要求するファイルの担当ディレクトリサーバを特定する必要がある。クライアント2000は、ファイルとそのファイルの担当ディレクトリサーバとの対応が記録されているファイル割当て管理表2001を参照し、取得を要求するファイルの担当ディレクトリサーバを特定する。次に、特定した担当ディレクトリサーバにファイル取得要求を送信する。

【0095】

ディレクトリサーバ2100は、クライアント2000からのファイル取得要求を受け取ると、ディレクトリ情報を参照して、該ファイルが格納されているストレージ2300を特定する。次に、特定したストレージ2300にファイルを要求する。ファイルの要求を受け取ったストレージ2300はファイルを送信し、ディレクトリサーバ2100はファイルを取得する。取得したファイルは、要求を送信したクライアント2000に対して送信される。

【0096】

ここで、ディレクトリサーバ2100のキャッシュについて説明する。

【0097】

一般に、ディレクトリサーバはキャッシュメモリ等のキャッシュを持っている。ディレクトリサーバ2100は、一度ストレージ2300から取得したファイルをメモリ上のキャッシュ領域に格納する。以降、ディレクトリサーバ2100は、同一のファイルの取得要求に対しては、キャッシングしたファイルを送信し、ストレージ2300からファイルを取得する処理を省略する。このようにすることで、ディレクトリサーバ2100からファイルを取得する処理の効率が向上する。

【0098】

ここで、第1の実施の形態のようにディレクトリサーバ・クラスタ2600を構成するディレクトリサーバの数を動的に変更する場合を説明する。

【0099】

追加ディレクトリサーバ2200がディレクトリサーバ・クラスタ2600に追加された場合、既存ディレクトリサーバ2100が担当しているストレージ2300のファイルのうち、いくつかのファイルの担当が追加ディレクトリサーバ2200に移管される。すなわち、いくつかのファイルの担当ディレクトリサーバが既存ディレクトリサーバ2100から追加ディレクトリサーバ2200に変更される。

【0100】

ストレージ2300のファイルの担当ディレクトリサーバが変更されると、これに伴って、クライアント2000が持つファイル割当て管理表2001の内容が変更される。具体的には、例えば、ファイル“File1”の担当ディレクトリサーバがサーバ“SRV1”からサーバ“SRV2”に変更される。この場合、全てのクライアント2000が持つファイル割当て管理表2001の内容が更新される。このファイル割当て管理表2001の更新は、第1の実施の形態と同様に管理サーバを設け、管理サーバから各クライアントに通知してもよいし、ディレクトリサーバ2100からクライアント2000に対して直接通知してもよい。

【0101】

次に、この新たに追加された追加ディレクトリサーバ2200のキャッシュの状態について説明する。追加ディレクトリサーバ2200をディレクトリサーバ・クラスタ2600に追加し、いくつかのファイルの担当ディレクトリサーバ2100が追加ディレクトリサーバ2200に変更された時点では、追加ディレクトリサーバ2200のキャッシュにはファイルが一切キャッシングされていない。そのため、担当ディレクトリサーバが変更されたファイルにファイルの取得要求があった場合には、要求されたファイルを一度ストレージ2300に要求する必要がある、キャッシュが効果的に働かない。そのため、該ファイル取得要求に対する返答には多大な時間がかかる。

【0102】

したがって、多数のファイルの担当ディレクトリサーバを一度に追加ディレクトリサーバ2200に変更した場合は、追加ディレクトリサーバ2200での処理が滞り、結果として、ディレクトリサーバ・クラスタ2600からクライアント2000への返答が全体的に悪化する。このようなシステムの性能悪化を防止するために、追加ディレクトリサーバ2200が担当するファイルを一時に一度に増やすのではなく、徐々に増やす必要がある。

【0103】

既存ディレクトリサーバ2100から追加ディレクトリサーバ2200に、ファイルの担当を徐々に移管する方法について説明する。この移管は、クライアント2000が持つファイル割当て管理表2001の変更、非特許文献1のようにハッシュ関数を用いることで実現できる。

【0104】

図12はファイル割当て管理表2001を変更する方法を模式的に示した説明図であり、ファイルの担当を徐々に移管する方法の一例を示す。このファイル割当て管理表200

1 は、ハッシュ関数 2002 と、サーバ名変換表 2003 から構成される。

【0105】

クライアント 2000 は、ファイル取得要求を送信する際に、まず、ファイル名をハッシュ関数 2002 に入力し、ファイル名をハッシュ値に変換する。なお、ここで変換されたハッシュ値の最大値は、ディレクトリサーバの総数よりも充分大きな値になるようにハッシュ関数を設定する。次に、変換したハッシュ値からサーバ名変換表 2003 を引くことで参照し、ハッシュ値とディレクトリサーバ名との対応を取得し、そのファイルの担当ディレクトリサーバ名を特定する。

【0106】

ここで、追加ディレクトリサーバ 2200 が担当するファイルを徐々に増やすには、サーバ名変換表 2003 を徐々に変更する操作で実現できる。具体的には、例えば、既存ディレクトリサーバ 2100 が担当していたファイル“File1”の担当ディレクトリサーバを追加ディレクトリサーバ 2200 へ変更する場合、ファイル“File1”のハッシュ値に対応するサーバ名変換表 2003 のエントリに記録されている既存ディレクトリサーバ 2100 を追加ディレクトリサーバ 2200 に変更する。この変更操作を、一度に複数のハッシュ値に対して行うのではなく、段階的に徐々に行う。

【0107】

この、既存ディレクトリサーバ 2100 から追加ディレクトリサーバ 2200 に、担当ディレクトリサーバを変更するファイルを増やす方法については、前述の第 1 の実施の形態と同様である。すなわち、図 1 の負荷制御プログラム 400 と同等の働きをするプログラムが各クライアント 2000 上で動作し、担当ディレクトリサーバが変更されるファイルの数を、負荷制御関数 411 によって算出することで徐々に増加させる。

【0108】

本発明の第 2 の実施の形態のストレージ・システムでは、前述したように、クライアント 2000 がファイルの取得を要求するディレクトリサーバ・クラスタ 2600 の構成が変更され、新たに追加ディレクトリサーバ 2200 が追加されたときは、該追加ディレクトリサーバ 2200 が担当するストレージ 2300 のファイルの数を、既存ディレクトリサーバ 2100 と比較して小さくする。このようにすることで、追加ディレクトリサーバ 2200 での処理が滞ることを防ぐことができ、ディレクトリサーバ・クラスタ 2600 全体としての処理の効率を高めることができる。

【産業上の利用可能性】

【0109】

本発明は、Web サーバ→アプリケーションサーバ→データベースサーバのように階層構造に従って処理される階層 Web システムにおける、前段サーバと後段サーバとの負荷分散に適用すると有用である。また、複数のディレクトリサーバが複数のストレージ 2300 のディレクトリ情報を分担して保持するストレージ・システムのディレクトリサーバにおける負荷分散に適用すると有用である。

【図面の簡単な説明】

【0110】

【図 1】 本発明の第 1 の実施の形態のクライアント・サーバシステムの概要を表すブロック図である。

【図 2】 本発明の第 1 の実施の形態の負荷制御プログラム 400 構造及び処理の流れを示すブロック図である。

【図 3】 本発明の第 1 の実施の形態の負荷制御関数 411 の例を示すグラフである。

【図 4】 本発明の第 1 の実施の形態の負荷制御関数 411 の他の例を示すグラフである。

【図 5】 本発明の第 1 の実施の形態の台数検出機能 401 の処理を示すフローチャートである。

【図 6】 本発明の第 1 の実施の形態の性能検出機能 402 の処理を示すフローチャートである。

【図 7】本発明の第 1 の実施の形態の負荷重み計算機能 4 0 4 の処理を示すフローチャートである。

【図 8】コネクション・プールの実装例を示すブロック図である。

【図 9】本発明の第 1 の実施の形態の負荷分散機能 3 0 0 データ構造、及びデータの処理を示すブロック図である。

【図 1 0】本発明の第 1 の実施の形態の負荷分散機能 3 0 0 の処理を示すフローチャートである。

【図 1 1】本発明の第 2 の実施の形態のストレージ・システムの構成を示したブロック図である。

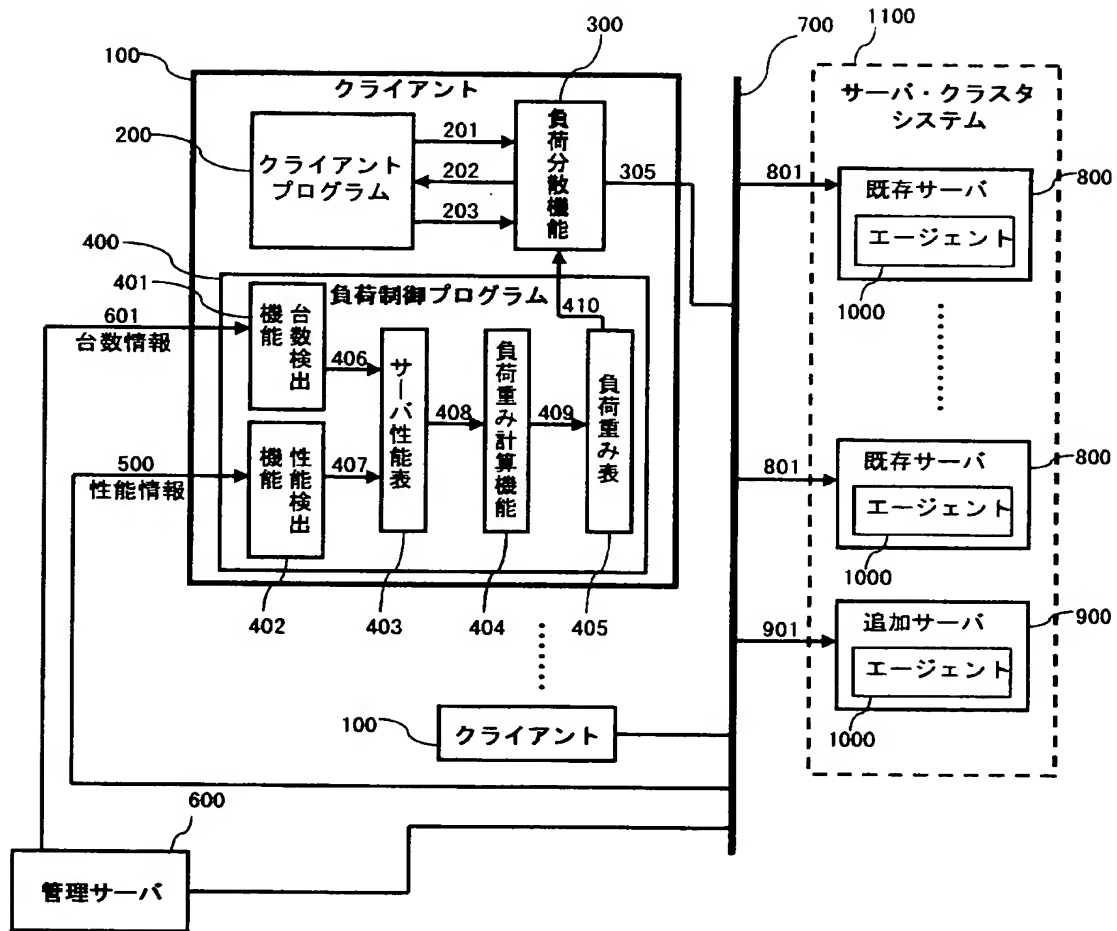
【図 1 2】ファイル割当て管理表 2 0 0 1 を変更する方法を模式的に示した説明図である。

【符号の説明】

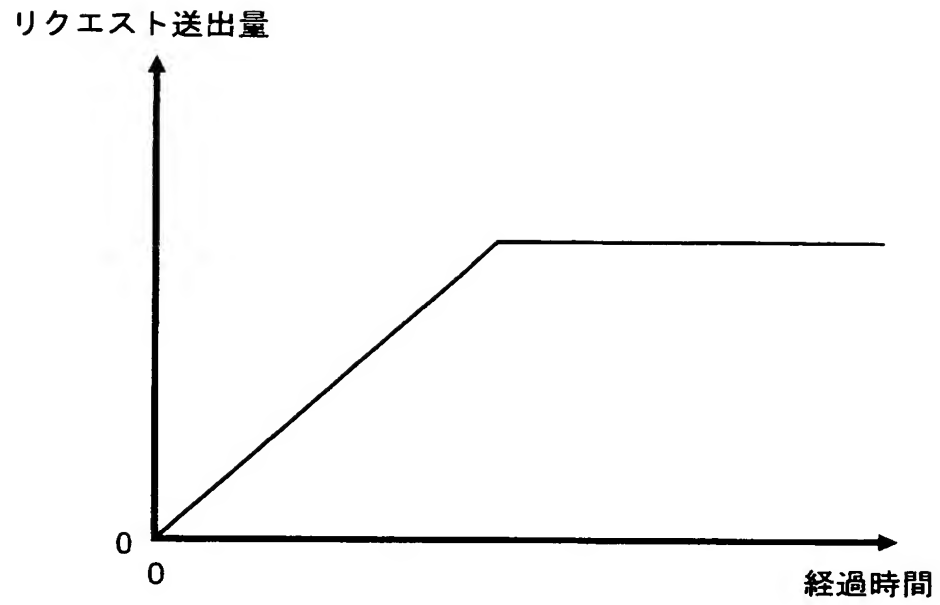
【 0 1 1 1 】

- 1 0 0 クライアント
- 2 0 0 クライアントプログラム
- 3 0 0 負荷分散機能
- 3 0 1 コネクション配分機能
- 3 0 2 コネクション管理表
- 4 0 0 負荷制御プログラム
- 4 0 1 台数検出機能
- 4 0 2 性能検出機能
- 4 0 3 サーバ性能表
- 4 0 4 負荷重み計算機能
- 4 0 5 負荷重み表
- 4 1 1 負荷制御関数
- 5 0 0 性能情報
- 6 0 0 管理サーバ
- 7 0 0 サーバ間ネットワーク
- 8 0 0 既存サーバ
- 9 0 0 追加サーバ
- 1 0 0 0 エージェント・プログラム
- 1 1 0 0 サーバ・クラスタシステム
- 2 0 0 0 クライアント
- 2 0 0 1 ファイル割当て管理表
- 2 0 0 3 サーバ名変換表
- 2 1 0 0 既存ディレクトリサーバ
- 2 1 0 1 エージェント
- 2 2 0 0 追加ディレクトリサーバ
- 2 3 0 0 ストレージ
- 2 4 0 0 L A N
- 2 5 0 0 S A N
- 2 6 0 0 ディレクトリサーバ・クラスタ

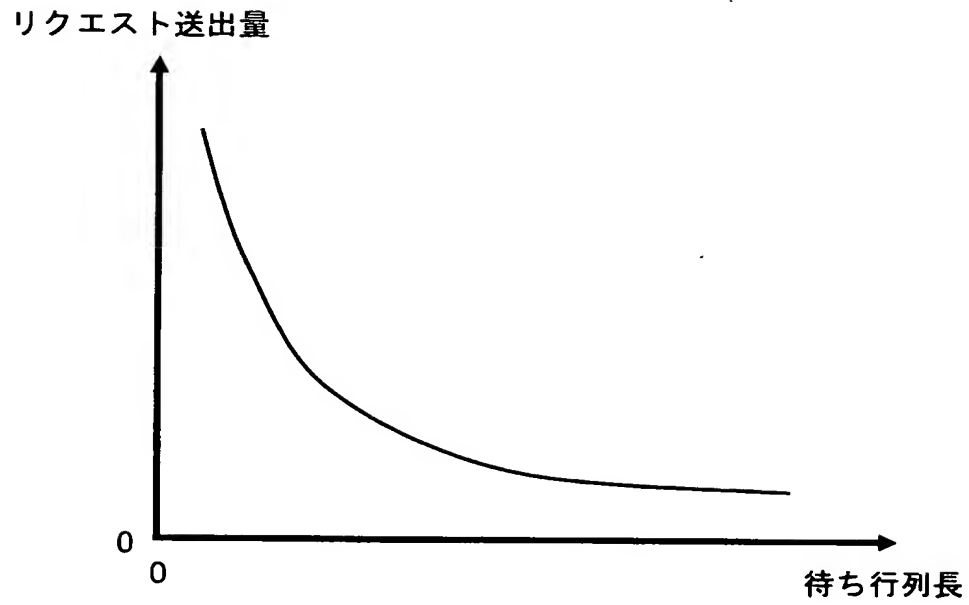
【書類名】 図面
【図 1】



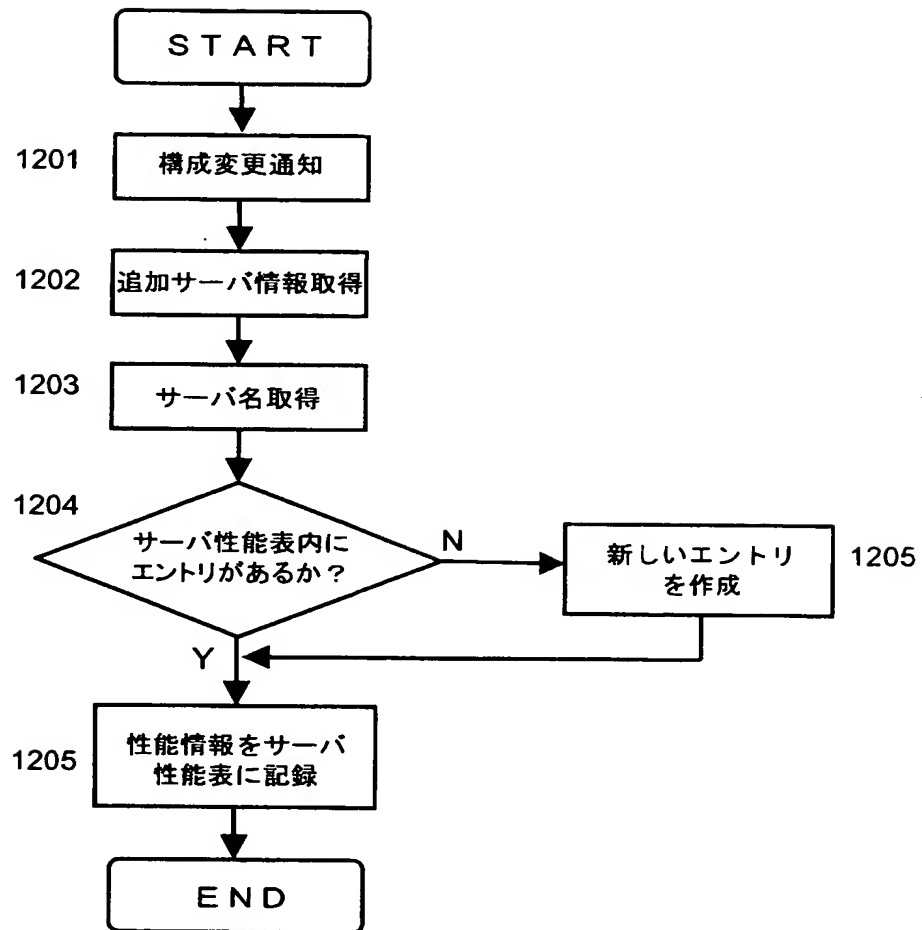
【図 3】



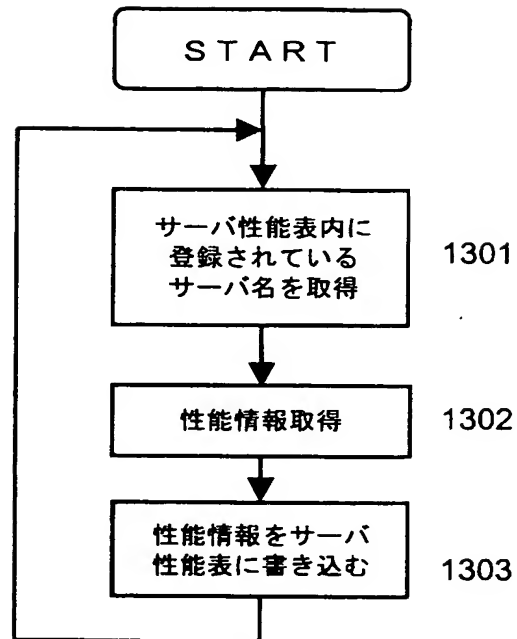
【図 4】



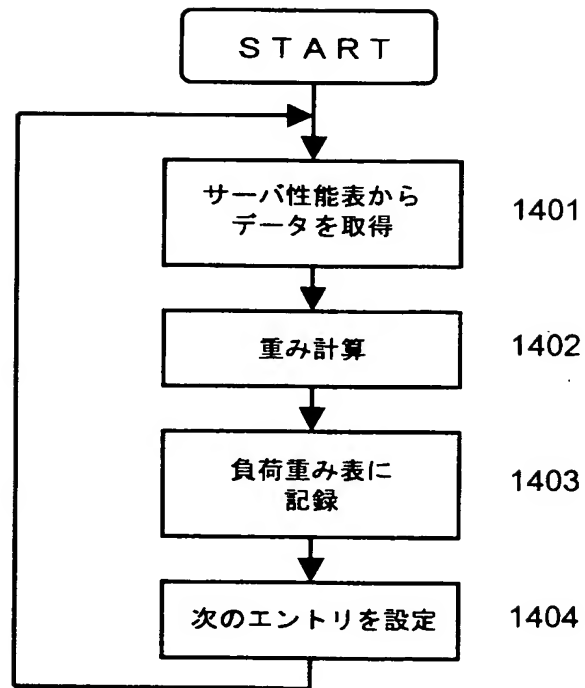
【図 5】



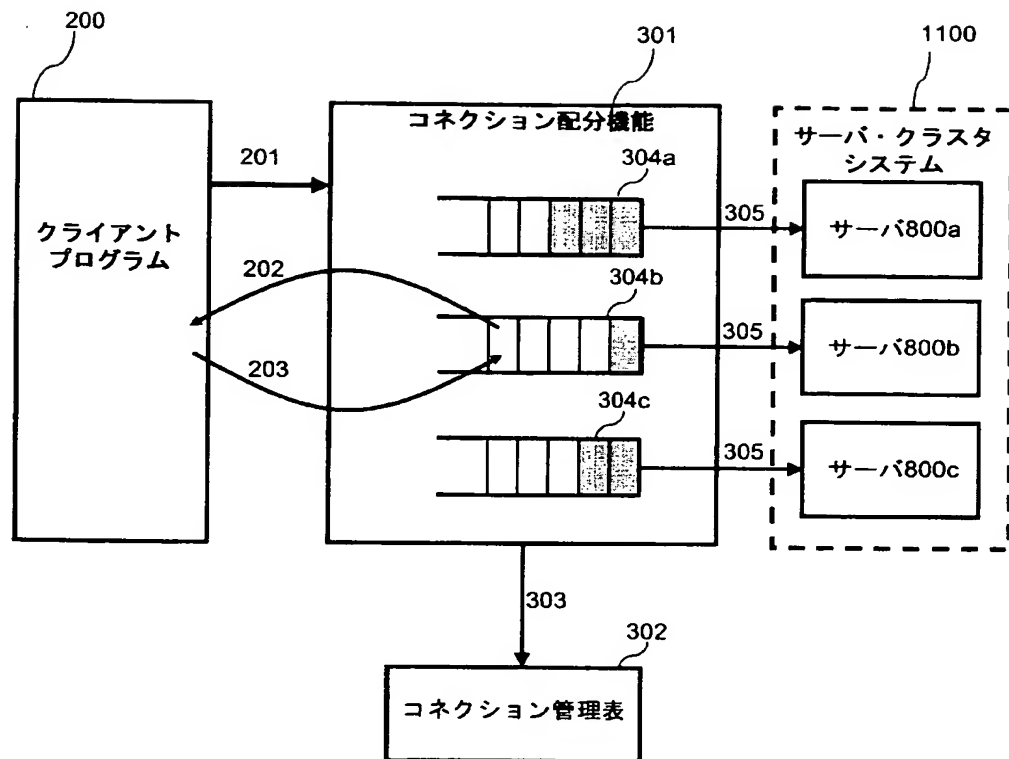
【図 6】



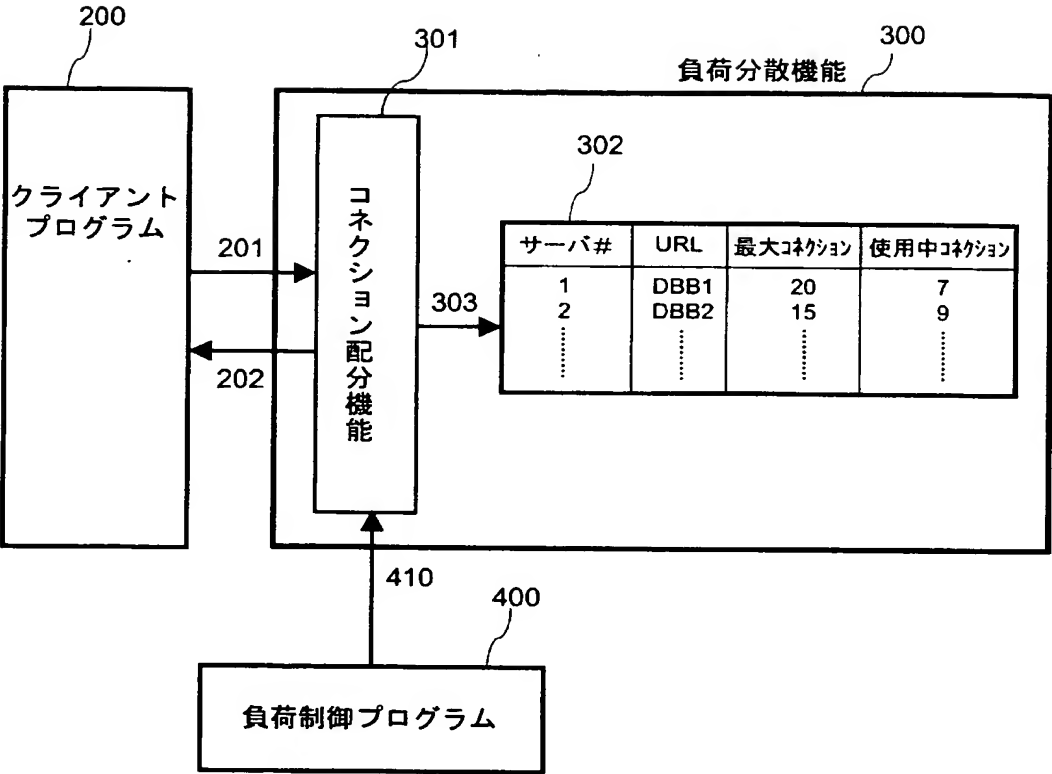
【図 7】



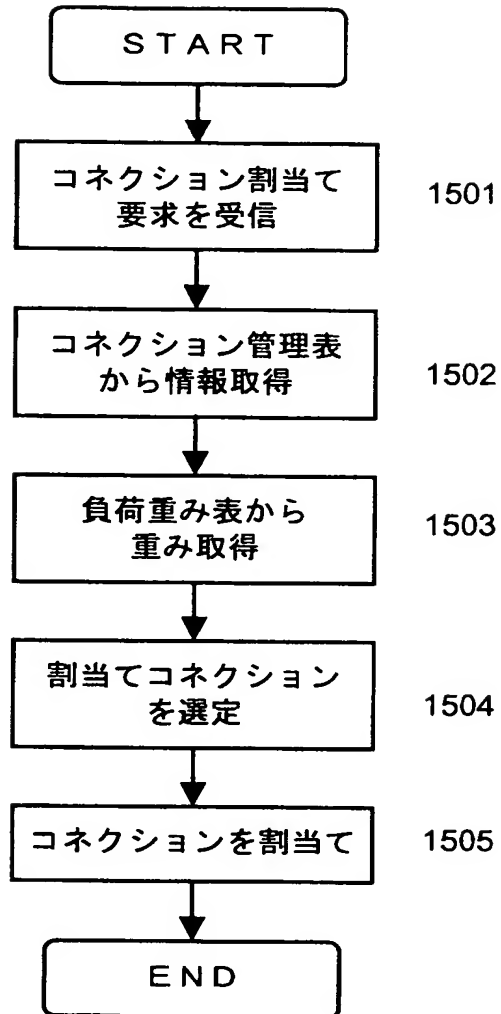
【図 8】



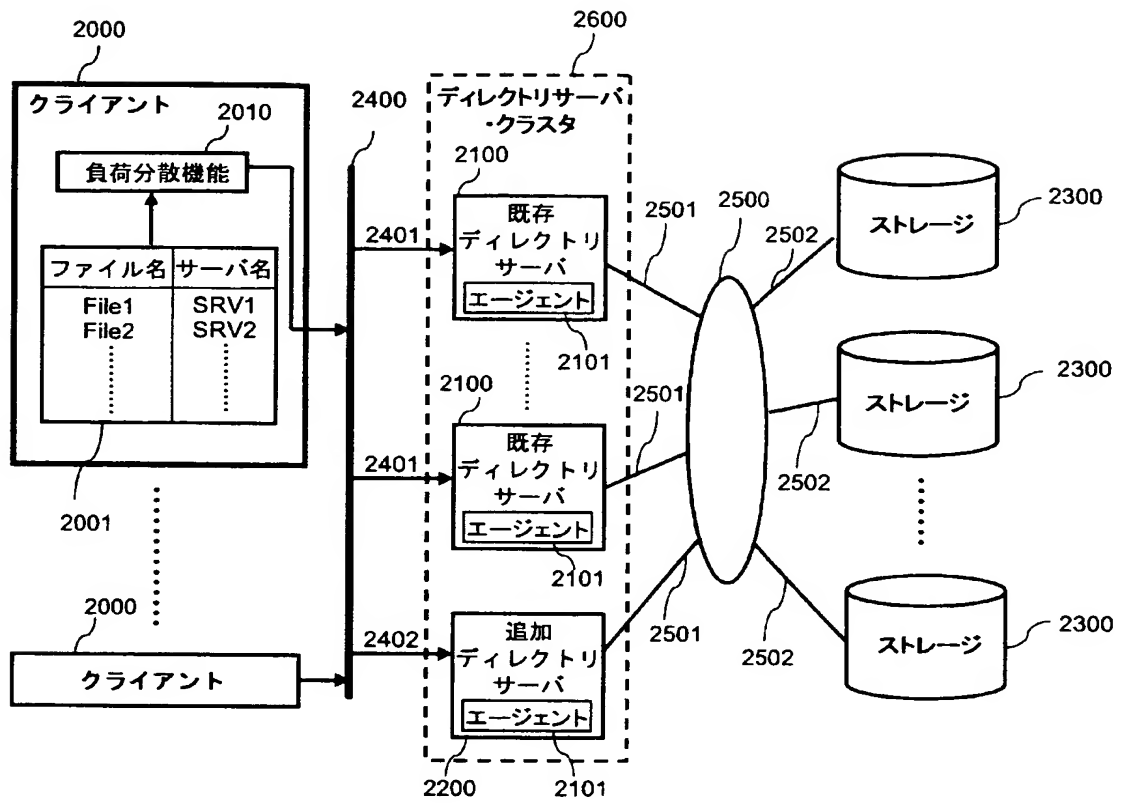
【図 9】



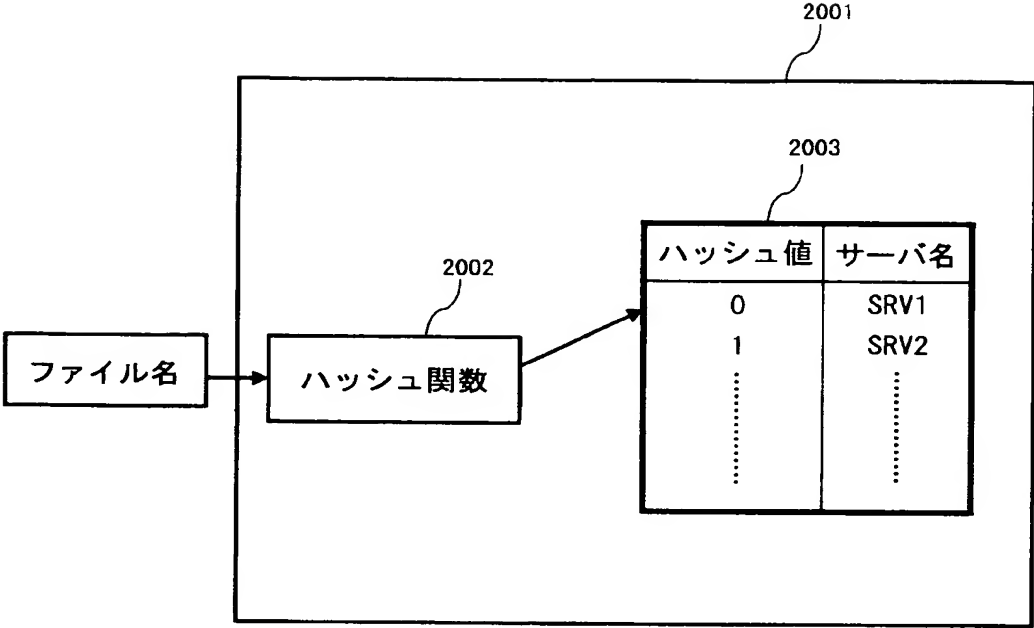
【図 10】



【図 11】



【図 1 2】



【書類名】 要約書**【要約】**

【課題】 サービス需要の増減に応じてサーバ・クラスタシステムを構成するサーバの台数を変更するクラスタ再構成技術に適合した、クライアントとサーバ・クラスタシステム間の負荷分散方法を提供する。

【解決手段】 複数のクライアント100と、クライアント100からのリクエストを処理する複数のサーバ800を含み、複数のサーバの数を動的に変更するサーバ・クラスタ1100と、によって構成されるクライアント・サーバシステムに用いられる負荷分散方法であって、クライアント100は、サーバ・クラスタ1100を構成するサーバの数を検出し、サーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定し、前記設定された配分に基づいて前記複数のサーバに対してリクエストを送出する。

【選択図】 図1

特願 2 0 0 3 - 2 9 1 2 8 6

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 5 1 0 8]

1. 変更年月日

1 9 9 0 年 8 月 3 1 日

[変更理由]

新規登録

住 所

東京都千代田区神田駿河台 4 丁目 6 番地

氏 名

株式会社日立製作所